Radiation Mitigation and Power Optimization Design Tools for Reconfigurable Hardware in Orbit

Matthew French[1], Paul Graham[2], Michael Wirthlin[3], Li Wang[1] and Gregory Larchev[4]
[1]University of Southern California, Information Sciences Institute, Arlington, VA
[2]Los Alamos National Laboratory, Los Alamos, NM
[3]Brigham Young University, Proto, UT
[4] QSS Group Inc., NASA Ames Research Center, Moffett Field, CA

Abstract- **The Reconfigurable Hardware in Orbit (RHinO) project is focused on creating a set of design tools that facilitate and automate design techniques for reconfigurable computing in space, using SRAM-based field-programmable-gate-array (FPGA) technology. In the second year of the project, design tools that leverage an established FPGA design environment have been created to visualize and analyze an FPGA circuit for radiation weaknesses and power inefficiencies. For radiation, a single event Upset (SEU) emulator, persistence analysis tool, and a half-latch removal tool for Xilinx Virtex-II devices have been created. Research is underway on a persistence mitigation tool and multiple bit upsets (MBU) studies. For power, synthesis level dynamic power visualization and analysis tools have been completed. Power optimization tools are under development and preliminary test results are positive.**

I. Introduction

SRAM-based FPGAs have become a promising solution to processing on space-based payloads. They offer features that anti-fuse FPGAs do not, such as reprogrammability, embedded multipliers, and embedded processors, while also offering 5-10x more logic gates. These features allow SRAM-based FPGAs to address resource multiplexing, fault tolerance, mission obsolescence and design flaws in on-orbit payloads that directly impact design cost and mission risk, while also providing better processing performance. However, a significant barrier to developing space-ready SRAM-based FPGA applications is the difficulty in designing for the rigorous constraints mandated by the operational environment. Two main issues limit the use of conventional FPGAs to such designs: 1) SRAM-based FPGAs are sensitive to radiation effects, namely, total ionizing dose (TID), single event latchup (SEL), and single-event-upsets (SEUs), because of their high proportion of memory structures; and 2) SRAM-based FPGAs designs tools optimize for throughput at the expense of power.

Current foundry process technology for Xilinx FPGA devices provides enough tolerance for a large number of ESE orbits for total dose and latch up (no destructive latchups have been reported), however the SEU presence is a major design/operational issue. The large amount of static memory within SRAM-based FPGAs, such as look-up tables, routing switch tables, etc., makes them sensitive to SEUs. While traditional hardware redundancy techniques improve the reliability of FPGA designs (at the expense of increases in hardware, power, etc.), novel FPGA-specific techniques are required to address the unique vulnerabilities of SRAM-based FPGA architectures, while incurring less hardware overhead. Therefore, design automation tools evaluating and assessing the reliability of FPGA designs, inserting appropriate redundant hardware, and manipulating the low-level structures of the FPGA design are needed for robust operation and SEU and latch-up tolerance.

Available FPGA synthesis tools optimize for speed or area, but not for real-time power consumption. Limited power estimation tools are available, such as Xilinx's XPower; however, these are difficult to use and have limited utility to the actual FPGA design process. Accurate power estimates are only achievable after completing an entire iteration of the design cycle and provide no power optimization guidance. To make effective use of FPGAs in space, tools providing accurate power estimation and dynamic power optimization, operating on the FPGA's gate logic or on individual configurable logic blocks (CLBs), are needed; specifically: 1) to monitor power consumption early in the design process at a useful granularity (e.g., at CLB); 2) to aid in the design analysis that captures data-dependent transients as well as overall power consumption; and 3) to perform automated dynamic power optimization.

Both the radiation-induced and power consumption effects are currently handled through manual intervention or, at best, through ad-hoc in-house tools. There is a real need in the community for validated design tool automation to raise the technology readiness level (TRL) of SRAM-based FGPA user designs. The RHinO project is leveraging an established, open-source tool-suite that accepts output from commercially available synthesis tools to create tools that allow the developer of a space-based FPGA application to automatically analyze and optimize a Xilinx Virtex II FPGA circuit for both space radiation effects and power utilization.

In the second year of this effort, the space radiation effects and power analysis tools were refined and leveraged to build optimization tools. The final year of this effort will focus on validating and improving the optimization tools in the relevant environment. This paper will outline the JHDL tool suite and extensions made to it for the RHinO toolkit in section II. SEU effects tools are discussed in Section III, and power utilities are discussed in Section IV. Synergy with evolvable algorithms for mitigating SEL effects are discussed in Section V. Section VI will summarize the progress to date and draw conclusions.

II. The JHDL Tool Suite

A. Background

As outlined in [1], the RHinO tools suite is built upon the open-sourced JHDL [2] FPGA design environment. The tool suite, shown in Figure 1, contains a digital circuit simulator, a circuit hierarchy browser, FPGA library primitives, and tools for exporting user designs into EDIF and VHDL. JHDL provides an open API into the circuit structure to facilitate the creation of application-specific design aids for viewing, revising, manipulating, or interacting with a user design. The integrated design aids, circuit API, and flexibility of JHDL make it an ideal tool for aiding the development of radiation-hardened and power-aware space-based FPGA designs. A variety of application-specific tools can be created to analyze and improve the reliability of FPGA circuits.

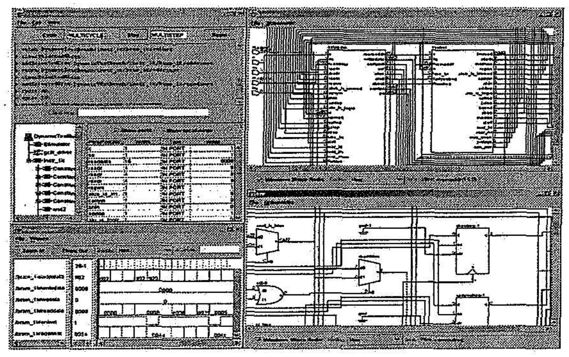


Figure 1. JHDL Tool Suite

Under this effort, RHinO is devising new features for JHDL, specific to space environments, which would enable SRAM-based FPGA payload developers to confidently manage the limiting on-board spacecraft design constraints for power, radiation effects, fault-tolerance, reliability, etc. A key goal of the effort is interoperation with existing commercial tool flows based on VHDL/Verilog, through seamless JHDL-EDIF translation. Alternatively, the user can work entirely in the JHDL design environment, using the RHinO power and SEU tools in concert with the normal JHDL features for simulation, netlisting, and runtime control, all within a single user interface.

B. RHinO Enhancements

During the first year of this effort the JHDL infrastructure was enhanced to support the desired SEU mitigation and power tool functionality. A GUI event API was developed to support intercommunication and interoperability with other modules, or tools that could be dropped into JHDL. As shown in Figure 2, this has led to the development of multiple tool modules being able to leverage the core JHDL capabilities.

Recently, considerable effort was spent enhancing the EDIF netlist tool, originally created to support importing 3rd party IP. The EDIF netlist parser and data structure software provides the central design database for both the RHinO power analysis tools and RHinO design reliability and mitigation tools. These tools provide two important capabilities for the RHinO tool suite. First, these tools provide the capability of *importing* an FPGA design created with a third-party tool into the RHinO infrastructure. Second, these tools provide a consistent circuit database for each of the tools created in the RHinO project.

The relationship between the EDIF tools and other RHinO tools is shown below in Figure 2. An FPGA design is loaded into the RHinO suite through the EDIF parser and into the EDIF data structure. At this point, the design can be manipulated or analyzed using one of several RHinO tool components. For example, power estimates of the design can be made by using the JHDL/RHinO power estimator tool chain. In this mode, a dynamic simulation of the design is created in JHDL to obtain the activity rates of design components and nets. The power estimation and viewer tools are available for browsing and viewing the results of this design simulation. Alternatively, the design reliability analysis tools may be invoked from the EDIF data structure. With these tools, the reliability of the design can be analyzed and presented to the user. As the project matures, design mitigation and power optimization techniques can be applied to *improve* the design over its original specification.

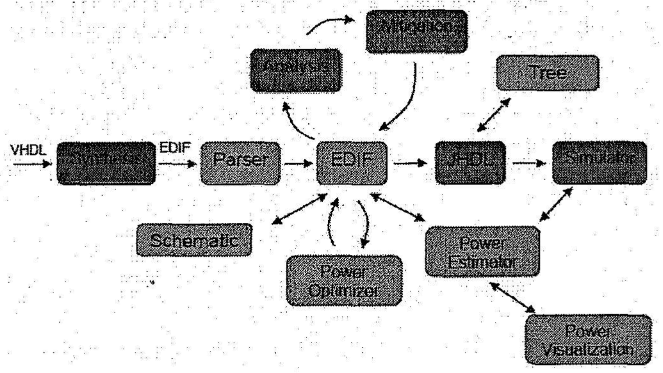


Figure 2. Tool Infrastructure

C. Cross Tool Naming and Correlation

An important capability required within this design infrastructure is the ability to correlate design resources between the various tool stages and with the COTS tool flow. For example in power analysis, design resources are represented in three different design databases. First, design resources are specified in the original EDIF source and are captured within the EDIF design environment. Second, the EDIF design is translated to the JHDL for dynamic simulation. Third, the EDIF design is translated to the FPGA technology specific netlist after technology mapping. The power capacitive loading values are made available at this level, but need to be relayed to higher levels to make power a first class design constraint.

To properly analyze and estimate the power consumption of FPGA designs, design resources must be correlated between the three design representations. Figure 3 below represents the relationship between these design representations. The difficulty in name correlation occurs due to the difference in the way that the vendor specific names and the JHDL simulation environment interpret names. The vendor-specific "Xilinx" naming scheme uses the EDIF "original" name for naming each of these resources. These names are the names chosen for design resources by the original design synthesis tool. JHDL uses the "valid" EDIF name to represent each of its design resources. A name management resource was added (represented by the red arrow to the right) to provide a fast matching capability between the simulation environment in JHDL and the capacitive loading resources defined by Xilinx.

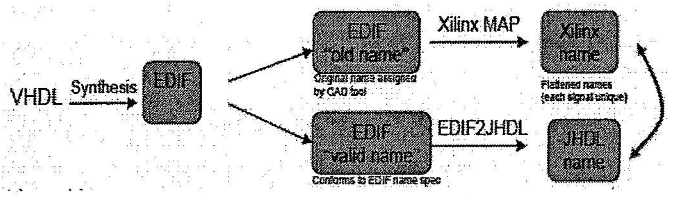

VHDL
Synthesis
EDIF
EDIF "old name"
Original name assigned by CAD tool
Xilinx MAP
Xilinx name
Flattened names (each signal unique)
EDIF "valid name"
Conforms to EDIF name spec
EDIF2JHDL
JHDL name

Figure 3. Signal Naming Relationships

These tool enhancements were critical to better interoperation with the commercial-off-the-shelf (COTS) synthesis, placement and routing (PAR) tools, as well as to provide what until now has been low-level bit-stream information at higher levels. Bringing this information to higher levels allows the tools outlined in sections III and IV to operate on smaller, faster files and databases, and more easily relay information to the user at the design entry level.

III. SEU Radiation Effects

A. Background

To further advance the TRL level of Virtex-II FPGAs for space applications, the RHinO project has a goal of improving the reliability of user designs in the presence of SEUs. SEUs are the main radiation concern since these FPGAs have been shown to have acceptable tolerance to TID as well as to SEL for low earth orbits (LEO). SEUs can occur in several memory structures on these SRAM-based FPGAs [3 4], namely in the support and control logic, the user design state, the programming memory (often called the *configuration memory*), and half-latches. Upsets in the support and control logic can have a range of effects, from fairly benign to totally erasing the contents of the FPGA configuration memory. Upsets in the user design state, such as in flip-flops and on-chip memories, may cause faults to occur in the user design's operation, while upsets in the configuration memory may change the user's design directly by changing the connectivity of logic or changing the logic functions themselves.

This year, the SEU radiation research completed the development of the SEU emulator and half-latch mitigation tool and focused on two specific areas, error persistence and multi-bit upsets (MBUs). Error persistence refers to length of time an error persists once a circuit experiences an SEU. Circuits with feedback paths or containing state machines may experience a significant degree of error persistence. For some applications, it may be enough to apply triple-modular redundancy (TMR) to the parts of the circuit exhibiting the most error persistence, drastically reducing the amount of overhead usually associated with TMR. MBUs due to a single charged particle are important since they could potentially affect multiple modules in TMR and produce incorrect circuit values.

B. SEU Analysis and Emulation

To analyze an FPGA design for SEU robustness for half-latches, persistence, and MBUs, an effective SEU emulation platform, the Virtex-II SEU Emulator (V2SE), was completed for characterizing individual designs for their particular set of SEU sensitivities. The V2SE is a combination of COTS hardware, custom software, and a single, low-cost (<$150) custom printed circuit board (PCB).

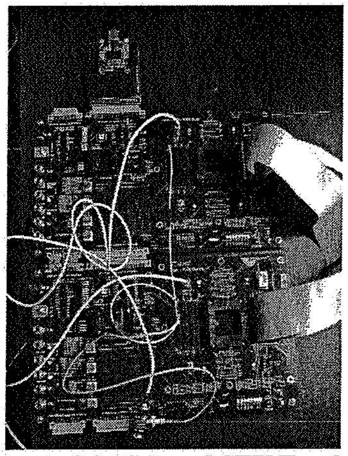


Figure 4. COTS-based SEU Emulator

The V2SE has been validated using a series of different tests and designs. Further, a graphical interface was developed to allow the user to better visualize the results of a particular execution of the V2SE on a design. Since the results of the SEU emulation sessions have the same floorplan as the user FPGA designs themselves, the visualization tools also suggest that SEU emulator is performing well.

The object of SEU emulation is to provide a cost-effective and quick alternative to accelerator testing for understanding user FPGA designs' SEU sensitivities. To validate this goal, the cost of upsetting 1.5 million programming bits through irradiation was compared against the costs of the V2SE. The V2SE, including the cost of a Linux PC to control the boards, costs about $6000. The V2SE can inject 1.5 million bit upsets in about 30 minutes given the above criteria. At an accelerator, a similar number of upsets with the same ability to distinguish the cause of output errors would take about 11 days of test time (assuming about 1.5 SEUs per second). Further, the V2SE provides considerably more control over what gets upset and when than the accelerator.

C. Half-Latch Mitigation

To meet the goals for automated SEU mitigation of Virtex-II user designs, the V2SE was employed in validating the RadDRC-II tool, which was developed to eliminate half-latch SEU sensitivities from user FPGA designs. Unlike the results of using SEU emulation for Virtex, SEU emulation for Virtex-II does not seem to upset half-latches as frequently. In fact, there was no noticeable difference between unmitigated Virtex-II designs and those mitigated by RadDRC-II.

The V2SE was then used for a proton radiation test at

Crocker Nuclear Laboratory to further validate the effects of the RadDRC-II. The initial results with 63 MeV protons suggest that RadDRC-II does improve the reliability of designs, but more test data is required to confirm this.

More specifically, two hard lockups of the unmitigated designs were observed—lockups that were typical of half-latch behavior with the earlier Virtex family. During one test, the unmitigated design also showed significant sequences of design lockups (>10 consecutive lockups). The mitigated designs did not show either of these behaviors.

D. Persistence

Characterizing the error persistence within FPGA designs [5 6] is of interest because it can lead to a way of gaining design robustness without extensive redundancy. For instance, for a certain test signal processing design, SEU emulation was performed on the design (i.e., intentionally injecting faults into the FPGA's programming data) and the FPGA programming bits that caused persistent errors were characterized. For this particular design, about 10% of the FPGA's configuration bits caused an output error while only about 0.24% of the configuration bits caused a persistent error—almost two orders of magnitude fewer bits (see Figure 4). This suggests that to have a design that continues to operate in the presence of SEUs without the need of a design reset, a significantly smaller amount of redundancy may be needed than full triple modular redundancy for the design.

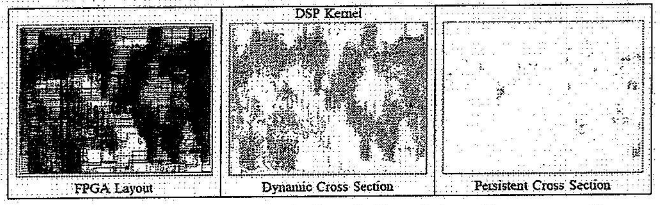


Figure 4. Comparison of Total Sensitive Cross-section (middle) to Persistent Cross-section (right) for a test DSP design (left)

During multiple tests at Crocker Nuclear Laboratory using 63 MeV protons, the SEU emulation predicting SEU error persistence for designs was verified. Four designs were tested, including the test digital signal processing (DSP) design (as mentioned above), a pipelined array of multipliers and adders, a design with an array of 8-bit counters, and a TMR version of the counter array design.

The bitstream SEU sensitivity observed at the accelerator matched the predicted sensitivity using SEU emulation quite well. For the unmitigated designs, greater than 90% of the bits that caused output errors in the accelerator testing also caused output errors during SEU emulation. With the TMR design we tested, only 60% of the bits that caused an output error seen by the accelerator were predicted by SEU emulation, but this was not a big concern since the design had very few problem bits to begin with at the accelerator (a total of 5), so the actual difference in only a few bits.

With regards to error persistence, the proton tests confirmed that SEU emulation is effective at predicting the statistics of the persistent errors. For the four test designs, the accelerator's results were within 15% of what was predicted through SEU emulation. Further, it was confirmed that there are no "hidden" structures in the FPGA (other than unmitigated half-latches) that will cause this persistence behavior.

The results also point out, though, that the prediction of a particular persistent failure can be very difficult due to the large state space for significant digital designs and the variable amount of time required for certain errors to be flushed out of the system. In other words, the persistent failures often depend heavily on when an SEU occurs, what was upset, and what data was being processed at the very moment of the upset. Further, for some designs, the longer one waits for values to flush out of the system, the fewer upsets will be classified as persisent since more and more will eventually flush out of the system. In other words, persistence as tested through SEU emulation and sampled at the accelerator is always relative to the amount of time allowed for errors to flush out of the system.

E .MBU Analysis

The main concern with respect to MBUs is whether or not MBUs can affect user FPGA designs that employ TMR or other forms of redundancy. To better understand the frequency of multi-bit upsets due to individual strikes by charged particles, Xilinx FPGAs from the Virtex, Virtex-II, and Virtex-4 families were irradiated with 63 MeV protons during multiple visits to Crocker Nuclear Laboratory. For these experiments, the occurrence of SEUs in these FPGAs was sampled and then the sampled SEUs were clustered based on their physical locations on the devices. An 8-bit neighborhood was used around each bit to determine adjacency. It was observed that 0.045% of the upset events experienced by Virtex XCV1000 FPGAs were MBUs while about 1.07% of all upset events experienced by Virtex-II XC2V250 and XC2V1000 FPGAs were MBU events—an increase in frequency of about 24 times between the FPGA families. As expected by Xilinx, Virtex-II exhibited a significant bias in the location of MBUs—88% of the MBUs were within the same configuration data frame—while Virtex MBUs were entirely within the same row of configuration data (adjacent configuration data frames with the same bit offsets). The Virtex-4 data has not been completely analyzed, but initial real-time feedback provided at the test suggested that about 1% of the events were MBUs.

For test data verification, the probability of causing false MBUs randomly was calculated and then compared against Monte Carlo simulations to verify these predictions. The results of this analysis are shown in Table 1. In the worst case, the actual observed MBU rates were more than 25 times the predicted false MBU rates, suggesting that an actual MBU phenomenon has been observed.

The RHinO team has also started analyzing heavy ion data that was taken by Xilinx and the Xilinx Radiation Test Consortium for Virtex-II. The early analysis of the data has been for LETs of 1.5-60 MeV/mg/cm^2. For these LETs, 1% to 35% of the events observed are MBUs. A more detailed report on MBUs in SRAM FPGAs can be found in [7].

FPGA	Bits in bit-stream	Upsets per read-back	Prob. of false MBU events (Analysis)	Prob of false MBU events (Monte Carlo)	Observed, Actual Rates	Ratio of Actual vs. False MBU Rates
4VLX25	7900864	750	0.0379%	0.0379%	1.000%	26.38
2V1000	3744768	280	0.0298%	0.0298%	1.070%	35.89
2V250	1588244	60	0.0149%	0.0149%	1.070%	71.83
V1000	5810048	10	0.000620%	0.000621%	0.045%	72.44

Table 1. Comparison of the probability of false MBUs with observed MBU rates

IV. Power

A. *Background*

Current SRAM-based FPGA design tools have been created with only speed or area optimizations as their goal and only recently have accurate power measurement tools become commercially available. These tools, such as the Xilinx XPower tool, are limited in the content of the power information they provide, readability, and their entry point in the FPGA design flow. Pre-place and route estimates are currently performed by manually entering and estimating device utilization, toggling rates, and routing interconnect, and automated power measurements are only available after going through a complete iteration of the entire design flow. This process can be ad hoc and time consuming, as a designer needs to interact with multiple tools that generate multiple memory-hungry intermediary files. At this level, the machine-generated signal names are difficult to resolve with their functional level counterparts in order to make optimizations. In other words, very little guidance is given to the designer on how to optimize if the power specifications were not met.

The goal of the power analysis and optimization tools is to make power a first-class design constraint by moving power analysis and optimization closer to the design entry point, with the help of the tool improvements outlined in Section II. Whereas the first year of this effort focused on developing accurate time-based power simulations of placed and routed circuits and enabling capabilities to rapidly sort, find, and cross probe signal and components, the second year used this infrastructure to develop more sophisticated power modeling to do power estimation and analysis at the post synthesis level, without the need for going through place and route.

B. *Post Synthesis Power Modeling*

Modeling at the post synthesis level has the following advantages: 1) early power feedback in the design flow, 2) power results are displayed at a high level, closer to the logical design entry point 3) and bulky, low-level timing accurate simulation and stimulus files are eliminated. These three aspects allow a designer to quickly and easily generate power estimates, relate the results back to their original logical level design entry, and explore design trade-off scenarios. The results presented here were derived using Xilinx Virtex-II FPGAs and tool suites, however the techniques apply to all FPGAs.

The technical approach can be broken into two parts, developing a tool infrastructure to support synthesis level simulation and circuit queries, and developing a synthesis-level power model. The tool infrastructure was discussed in section II to develop power tools to support querying and assigning capacitance values, and tracking wire toggle rates during simulation. Additionally, interoperability with the Xilinx tool flow was added, allowing XPower reports and .ncd files to be imported for detailed analysis of a placed and routed circuit's power characteristics. This environment allows a synthesized circuit to be directly compared against its placed and routed form to track what factors at the synthesis level lead to high power consumption at the placed and routed level. This environment allows the development and experimentation of several power models, from generic 'toggle rate only' models to the exact timing-level placed and routed circuit. After the power models have been created and verified, the end user can import their design into the JHDL environment using the EDIF import tool and simulate using the Synthesis Power Model (SPM), to generate synthesis level power reports, as depicted in Figure 5.

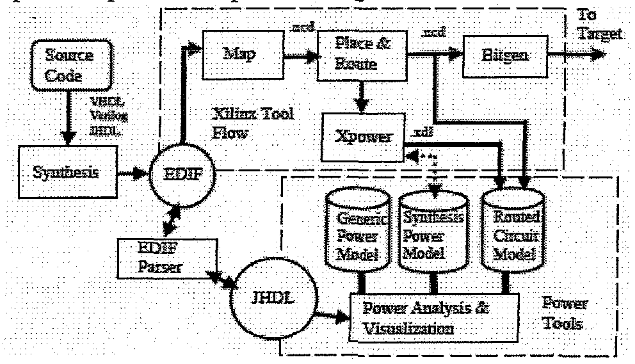


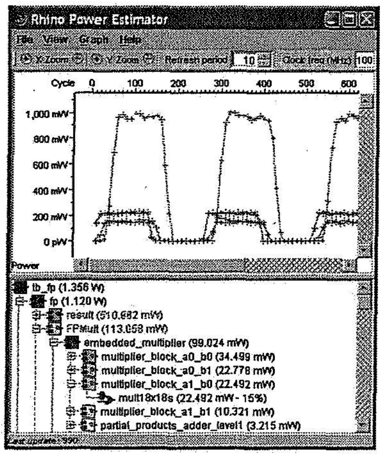
Figure 5. Power Modeling Tool Infrastructure

For generation of the SPM, placed and routed circuit power consumption was analyzed and then compared with the available information at the synthesis level. Dynamic power consumption is described as:

$$Power = \sum_i F_i * T_i * C_i * V_i^2 \qquad (1)$$

where F is the frequency, T is the toggling rate, C is the capacitance and V is the voltage of the ith component. For our modeling, we will assume the voltage will be fixed as per device specifications. The frequency and toggling rate of each net can be tracked within the JHDL simulation environment, allowing multiple subcomponents of the design to be tracked and the window of time that the power is averaged over to be changed dynamically during simulation, as shown in Figure 6.

The final term, the capacitance of each component, can be broken into two parts, the capacitance of the FPGA logic resource (ie LUT, BlockRAM, Multiplier etc) and of the interconnect route that it drives. Some capacitance information of FPGA resources has been published and other resources' capacitance values can be extracted from the XPower reports to create a complete resource capacitance model for any Xilinx device.



```
tb_fp (1.356 W)
fp (1.120 W)
  result (510.682 mW)
  FPMult (113.058 mW)
    embedded_multiplier (99.024 mW)
      multiplier_block_a0_b0 (34.499 mW)
      multiplier_block_a0_b1 (22.778 mW)
      multiplier_block_a1_b0 (22.492 mW)
        mult18x18s (22.492 mW - 15%)
      multiplier_block_a1_b1 (10.321 mW)
      partial_products_adder_level1 (3.215 mW)
```

Figure 6. Power Tool Simulation

The final and perhaps most difficult piece is modeling the routing interconnect. While an unoptimized list of the resources is known at after synthesis, the exact routing interconnect is unknown until placement and routing is performed, which can account for 50-70% of the dynamic power consumption of a circuit. In the Xilinx Virtex 2 device for example there are 4 types of interconnect (direct connect, doubles, hex, and long lines), each with a different capacitance and they can be joined in any number of combinations. Slight logic changes or different random seed inputs to the Xilinx place and route tools can yield large differences of a particular net's length as the router is purely timing constraint driven.

Using the JHDL power infrastructure, several designs on a Virtex-II 6000 varying in functionality and utilization, as shown in Table 2, were profiled and relationships between wire capacitance and fanout, wire length, number of switch boxes etc, were explored. Fanout proved to not only be a strong predictor of net capacitance; it also correlates well from the placed and routed circuit to the synthesized circuit. Though wire length and the number of switch boxes also correlated well with placed and routed net capacitance, this information is not available at the synthesis level. Figure 7 depicts the capacitance versus fanout plot for the nets in the Counter circuit. The fanout relationship was aggregated over all the designs in the circuit modeling pool to develop the SPM. It should be noted it is not expected to achieve precise power estimation at the post synthesis level, but rather to be accurate enough to guide the user to design-level hot-spots in the circuit. Single net outliers from the fanout behavior are best left to lower level analysis and optimization in XPower and FPGA Editor.

After the SPM was developed, we modeled the power consumption of the circuits in Table 1 and compared them to the values obtained by using the corresponding placed and routed circuit and XPower results. For the counter circuit a mean power error of 3.2% per net and load, with a standard deviation of 1.6% is achieved. Further enhancements to the model, such as predicting routing congestion based on total device utilization, are also being evaluated.

Design	Utilization	Flip-Flops	Multipliers
Test Circuit	2%	1,737	8
Config. Controller	4%	1,240	0
Network Interface	8%	3,083	0
Counters	10%	3,658	0
FP Multiplier	10%	4,243	52
Design 1	11%	5,254	12
DSP Soft Core	16%	1,981	0
Image Conv. Kernel	22%	7,231	18
8x8 Image Conv.	30%	11,293	130
FFT core	47%	23,098	96
Design 2	49%	23,291	84
Design 3	64%	19,389	120

Table 2. Circuit Power Modeling Pool

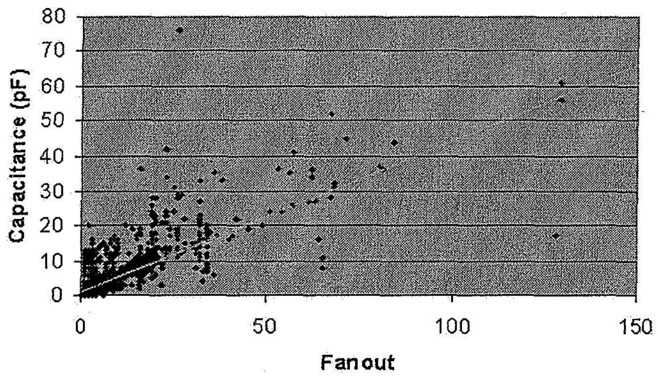


Figure 7. Net Capacitance vs. Fanout

C. Power Optimization

Dynamic power consumption can be reduced by reducing any term in equation 1, however as the voltage is fixed and the operating frequency and toggling rates are largely application driven, the most robust approach is to lower the switching capacitance. Dynamic power spans I/O power, logic component power, and signal power. The initial focus of the power optimization tools is to reduce the signal and clocking power of a circuit as I/O power is determined more by external interfacing requirements off chip, and logic component, such as CLBs, block RAMs, block multipliers, and DLL/DCMs are determined during synthesis.

At the level in the COTS tool flow that the RHinO tools are currently capable of addressing, the biggest power variable we can address is length, and therefore capacitance of the signal interconnects. In the following sections, two methods for optimizing signal power within the context of the COTS PAR tools without altering FPGA design functionality are discussed. Both of these methods work based on shortening signal routing paths for reducing signal capacitances.

Figure 8 depicts the power optimization tool flow on the left hand side, and the verification tool flow on the right hand side. The power optimization tools for both methods generate user constraint files, which are input to the Xilinx PAR tools. With these added constraints, the COTS tools are able to create more power efficient circuits. The first power optimization method presented is to add timing constraints on signals. The second method is to add location constraints on flip-flops. Both methods have the effect of over constraining power sensitive parts of the circuit, forcing the PAR tools to use shorter, lower capacitance interconnects.

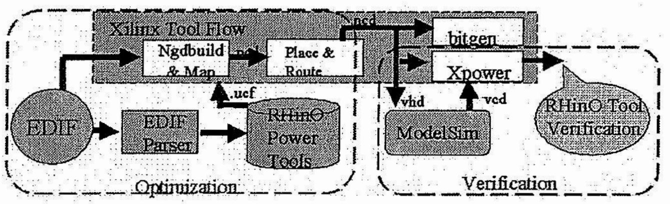


Figure 8. Power Tool Flow

Timing Constraint Based Power Optimization

In the timing constraint power optimization approach, timing constraints are added to signal wires to essentially translate power interconnect specifications into timing constraint specifications that the Xilinx PAR tools can work with. By raising timing constraints above the minimum operating frequency for high capacitance signals in a design, the PAR tools work to achieve shorter, or more power efficient, routes. From VLSI interconnect technology[8], the delay of a wire is proportional to wire capacitance *C*, therefore minimizing delays has the added benefit of minimizing capacitance and therefore power. In this way, power optimization is achieved by over-constraining the timing on signal wires to get lower signal capacitances.

Currently, the power optimization tools provide the user with a wire table, which can be sorted by simulated power consumption, load, fanout, etc. This helps the user to identify the most power critical wires, and rapidly create timing constraints on every net in a design if desired. For preliminary results to verify our methodology, we used a memory diagnostic and EDAC circuit for testing which contained 10,259 signals. The average toggling rate per wire during simulation was 12.5%. Testing was done on a variety of cases such as putting constraints on all toggling nets, constraints on nets with fanout of 10 or less, the top 25% most power hungry nets, etc. Preliminary results showed better placed and routed circuits with up to a 12% decrease in dynamic power consumption without changing the functionality of the circuit. Experiments are also underway to determine the effects of how much timing delay tightening should or can be applied.

Location Constraint Power Optimization

Another way to affect the commercial PAR tools is through specified location primitives that either define relative placement to other macros in a circuit or absolute placement within the device. By defining the placement with an eye towards power optimization, potentially high capacitance signal lines with high fanout or high toggle rates can be grouped together to minimize routing interconnect. Also, another important effect of this approach is that it can pare down the clock distribution tree, further reducing power.

Global clock signals in FPGAs have dedicated low-skew nets with short delays. In the Xilinx Virtex-II FPGAs, the clock nets are distributed like a tree into 4 clock zones: northwest, northeast, southwest, and southeast [9]. Furthermore, each clock zone quadrant branches further into sub-zones. In the chip investigated here, the Vertex-II 6000, each clock zone had 6 clock sub-zones, 88 x 16 slices in each sub-zone. The main clock trunk travels in the north south direction across the middle of the chip, with clock branches extending out from the trunk to the west and east into the sub-clock zones and clock sink cells. Clock sink cells include flip-flops, which dominate clock sink components, block RAMs, block multipliers etc. Unused clock branches remain static, so moving logic to trim the clock tree reduces power consumption. Figure 9 depicts two different placements of two flip-flops. On the right hand side, it can be seen that by placing two flip flops of the same clock domain near each other an entire quadrant of the clock tree can be trimmed, reducing power.

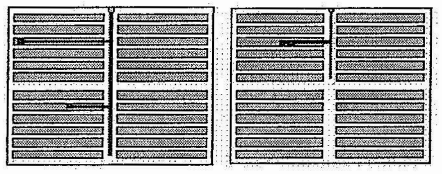


Figure 9. Logic Placement Clock Tree Effects

The power optimization tools for placement currently provides the mechanisms to help users decide where to put flip-flops and how many FPGA slices are need to accommodate a groups of flip-flops. The tool generates location constraints automatically. For the same test design used in the timing constraint experiments, there are 4 clocks, with fanout ranging from 488 to 2825. Preliminary test results were obtained for two cases. In the first case, the circuit was treated as a module, where I/O logic was free to be moved. In this case, dynamic power was reduced by up to 23%. For the second case, the I/O logic was left in specified IOBs, as in a typical full chip design. In this case the power was reduced by up to 11%.

D. Current Status and Future Direction

The preliminary power optimization results are encouraging. It should be noted that these tools currently are at a level where user guidance is still required for optimization and only a few of the possible optimization techniques have been experimented with. In the upcoming year, the research will focus on automating the optimization process through various algorithmic techniques. Also, the timing constraint approach and the placement approach are not mutually exclusive, so the effects of combining these two methods will also be explored.

Finally, the JHDL module generators will be expanded to include power (and SEU tolerance) as a design entry constraint, in addition to traditional throughput and precision constraints. The module generators can then use the known performance and capacitance information of FPGA components to create optimal modules for a particular design.

V. Evolvable Techniques

A. Background

Though SEUs are the primary radiation concern when dealing with SRAM-based FPGAs, for certain orbits and life-spans, TID and SEL do become a concern. Since it is virtually impossible to replace spacecraft components in-situ, there is a clear opportunity for fault-tolerant FPGA circuits.

Evolutionary algorithm (EA) methods hold promise in their ability to search across the space of FPGA configurations for those that can function in the presence of certain types of faults. Since SRAM-based FPGAs are fully reprogrammable, it is possible to restore the functionality of the compromised FPGA by re-routing a circuit around corrupted resources, a property which the RHinO team is exploring.

Autonomous repair can either provide an alternative to or supplement redundancy as a means of restoring lost capability. Some circuit configurations are more responsive to evolutionary repair than others. If a particular circuit has been shown to respond well to evolutionary repair, then EAs can be relied on as a primary source of fault tolerance. This allows the engineers to avoid the increased size, weight, and power consumption, traditionally associated with providing redundant spares. In cases when EAs have difficulties producing fully functional repairs, it is still possible to use these methods alongside traditional redundancy techniques. By repairing each individual triplet of a triple-modular-redundant system, it is possible to improve the performance of each triplet by a large enough margin so that the majority output is 100% correct (even if each individual output is not.)

The objective of the work described here was to investigate how various small circuits (some of which are commonly used in spacecraft electronics design) respond to a pre-determined level of simulated radiation damage. One sequential and three combinational circuits were tested. The sequential circuit was the quadrature decoder (a 4-state state machine.) The combinational circuits were a 3-by-3 multiplier, a 3-by-3 bit adder and a 4-to-7 bit decoder (a circuit used to control the individual segments of a 7-segment LED display.) The circuits were subject to a number of simulated faults, where at least 10% of the circuit's LUTs would be set to produce either a constant 0 or a 1 (simulating an output short to power or ground.) In addition, between 1.5% and 2% of all the LUT bits were "hard-wired" to 0 or 1. Such fault scheme was undertaken in order to try and take into account the actual logic vs. routing transistor distribution on an FPGA.

With the specified fault penetration, the average repair rate (percentage of circuits which achieved 100% repair) ranged from 0% (for the multiplier) to 90% (for the 4-to-7 bit decoder.) However, the average improvement in circuit performance over the course of each run ranged between 12.6% and 22.8%. What these results tell us is that while some circuits respond better to evolutionary repair than others, the EA methods result in noticeable improvement in performance in all the circuits tested. The chance of successful repair usually depends on the size and complexity of the circuit as well as the number and location of the faults; but even if EA approach cannot completely repair each individual circuit, there is a good chance that it can restore 100% overall functionality when used in conjunction with TMR.

B. Current Status and Future Directions

The evolutionary algorithm is being further refined to handle larger, more sophisticated circuits using the Virtex-II FPGA architecture, with the goal of showing that the algorithm is robust enough to solve SELs in the project's benchmark 3x3 image convolution kernel and other real-life applications. When a fault occurs in a large, complex circuit, the plan is to isolate the fault to a simpler component and then to re-evolve the component. Operating on full-sized applications on FPGA hardware will be a major thrust of this effort.

All of the team's FPGA evolution work to date uses bitstring chromosomal representation. This representation is the simplest but also the least efficient one. As larger circuits are considered, the shortcomings of bitstring representation will become more apparent. Therefore, work is underway to change the algorithm to a generative (tree-like) representation. By being more conducive to component reuse, generative representation shortens the chromosome length and makes the evolution of the larger circuits more manageable.

In the upcoming year, the algorithm will be proven on larger application circuits and integrated with the rest of the RHinO toolkit. Ultimately, the evolutionary algorithm will evolve circuits which are not only immune to existing SELs but also use guidance from the rest of the RHinO toolkit to create circuits that are SEU tolerant as well.

VI. Conclusions

In the second year of this effort, the baseline tool infrastructure was built upon enabling breakthroughs in radiation analysis tools and power analysis tools. The JHDL infrastructure was enhanced to allow better GUI development and fully support EDIF file import from a variety of commercial synthesis tools, which allowed multiple tools to be added to the JHDL backbone. In the radiation arena, the SEU emulator was completed, the half-latch tool was verified, and work has begun on persistence and MBU analysis. Finally, the power analysis modeling tools were completed, allowing a user to obtain accurate power estimations early in the design flow and power optimization tools have been developed, yielding promising initial returns. The next year's efforts seek to leverage the infrastructure, further, validating and verifying the tools and measuring the power trade-offs of various radiation mitigation schemes.

References

[1] French, et al., "Design Tools for Reconfigurable Hardware in Orbit," Earth Science Technology Conference 2004, Palo Alto, California, June 2004.

[2] "Adaptive Computing Systems"; 1997- 2003 DARPA effort; see www.jhdl.org

[3] Carl Carmichael, Earl Fuller, Phil Blain, and Michael Caffrey, "SEU Mitigation Techniques for Virtex FPGAs in Space Applications", Proceeding of the Military and Aerospace Programmable Logic Devices International Conference (MAPLD), Sept. 1999, Laurel, MD, pp. C2.1-8.

[4] Michael Caffrey, Paul Graham, Michael Wirthlin, Eric Johnson, and Nathan Rollins, "Single-Event Upsets in SRAM FPGAs", Proceedings of the 5th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD), Sept. 2002, pp. P8.1-6.

[5] E. Johnson, K. Morgan, M. Wirthlin, M. Caffrey, and P. Graham, "Persistent Errors in SRAM-based FPGAs," MAPLD '04, Washington, D.C., September 2004.

[6] K. Morgan, E. Johnson, B. Pratt, M. Wirthlin, M. Caffrey, and P. Graham, "SEU Induced Error Propagation in FPGAs," IEEE NSREC '05, Seattle, WA, July 2005, To be presented.

[7] P. Graham, H. Quinn, J. Krone, M. Caffrey, and S. Rezgui, "Radiation-Induced Multi-Bit Upsets in SRAM-Based FPGAs," IEEE NSREC '05, Seattle, WA, July 2005, To be presented.

[8] www.ece.utexas.edu/~adnan/vlsi-04/lec6Interconnect.ppt

[9] http://www.xilinx.com/bvdocs/publications/ds031.pdf, page 36.